Phase 3: Flood monitoring and early warning system Based on IOT.

**HC-SR04 Ultrasonic Distance Sensor:**

|  |  |
| --- | --- |
| **VCC** | **Voltage supply (5V)** |
| **TRIG** | **Pulse to start the measurement** |
| **ECHO** | **Measure the high pulse length to get the distance** |
| **GND** | **Ground.** |

**Operation**[**​**](https://docs.wokwi.com/parts/wokwi-hc-sr04#operation)**:**

**To start a new distance measurement set the TRIG pin to high for 10uS or more. Then wait until the ECHO pin goes high, and count the time it stays high (pulse length). The length of the ECHO high pulse is proportional to the distance. Use the following table to convert the ECHO pulse length in microseconds into centimeters / inches.**

**A piezoelectric buzzer:**

### Operation modes[​](https://docs.wokwi.com/parts/wokwi-buzzer#operation-modes):

**The buzzer can operate in two modes: "smooth" (the default) and "accurate".**

**"smooth" sounds better and is suitable for simple, single-frequency tones. Use it when playing a melody or playing tones with Arduino's tone() function. Complex and polyphonic sounds may not play correctly (or not play at all) in "smooth mode"**

**Use the "accurate" mode when you need to play complex sounds. It will accurately play the sound you feed in. However, it'll add audible click noises to your sound. These noises are due to fluctuations in the simulation speed - it's not always able to provide the complete sound buffer in real time**.

**Digital Humidity and Temperature sensor.**

|  |  |
| --- | --- |
| **VCC** | **Positive voltage** |
| **SDA** | **Digital data pin (input/output)** |
| **NC** | **Not connected** |
| **GND** | **Ground.**  **\*LED Bulbs**  **\*Resistor**  **\*Wire**  **Program:** |

**import time**

**import machine**

**import dht**

**TRIG\_PIN = machine.Pin(2, machine.Pin.OUT)**

**ECHO\_PIN = machine.Pin(3, machine.Pin.IN)**

**BUZZER\_PIN = machine.Pin(4, machine.Pin.OUT)**

**DHT\_PIN = machine.Pin(5)**

**LED\_PIN = machine.Pin(6, machine.Pin.OUT)**

**def distance\_measurement():**

**TRIG\_PIN.on()**

**time.sleep\_us(10)**

**TRIG\_PIN.off()**

**while not ECHO\_PIN.value():**

**pass**

**pulse\_start = time.tick**

**while ECHO\_PIN.value():**

**pass**

**pulse\_end = time.ticks\_us()**

**pulse\_duration = time.ticks\_diff(pulse\_end, pulse\_start)**

**distance = pulse\_duration / 58**

**return distance**

**def read\_dht\_sensor():**

**d = dht.DHT22(DHT\_PIN)**

**d.measure()**

**return d.temperature(), d.humidity()**

**buzz\_start\_time = None**

**while True:**

**dist = distance\_measurement()**

**temp, humidity = read\_dht\_sensor()**

**if dist > 50:**

**BUZZER\_PIN.on()**

**LED\_PIN.on()**

**status = "Flooding Detected"**

**buzz\_start\_time = time.ticks\_ms()**

**elif buzz\_start\_time is not None and time.ticks\_diff(time.ticks\_ms(), buzz\_start\_time) >= 60000:**

**BUZZER\_PIN.off()**

**LED\_PIN.off()**

**status = "No Flooding Detected"**

**else:**

**status = "No Flooding Detected"**

**print(f"Distance: {dist:.2f} cm")**

**print(f"Temperature: {temp:.2f}°C, Humidity: {humidity:.2f}%")**

**print("Status:", status)**

**time.sleep(2)**

**OUTPUT:**

**Distance: 260.59 cm**

**Temperature: 24.00°C, Humidity: 40.00%**

**Status: Flooding Detected**